ELSEVIER

Computational Geometry 15 (2000) 25–39

Computational Geometry
Theory and Applications

www. elsevier.nl/locate/comgeo

Progressive transmission of subdivision surfaces

U. Labsik [a], L. Kobbelt [b,*], R. Schneider [b], H.-P. Seidel [b]

[a] *Computer Graphics Group, Universität Erlangen-Nürnberg, Erlangen, Germany*

[b] *Computer Graphics Group, Max-Planck-Institut für Informatik, Stadtwald, 66123 Saarbrücken, Germany*

Abstract

Triangle meshes are a standard representation for surface geometry in computer graphics and virtual reality applications. To achieve high realism of the modeled objects, the meshes typically consist of a very large number of faces. For broadcasting virtual environments over low-bandwidth data connections like the Internet it is highly important to develop efficient algorithms which enable the progressive transmission of such large meshes. In this paper we introduce a special representation for storing and transmitting meshes with subdivision connectivity which allows random access to the detail information. We present algorithms for the decomposition and the reconstruction of subdivision surfaces. With this technique, the receiver can reconstruct smooth approximations of the original surface from a rather small amount of data received. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Level of detail management; 3D shape simplifications; 3D geometric shape modeling

1. Introduction

In computer graphics and virtual reality applications, triangle meshes have emerged as a common and most versatile free-form surface representation. For realistic objects it is often necessary that these meshes consist of a large number of faces. Using distributed data bases and remote digital libraries for archiving mesh models, it is getting highly important to be able to efficiently broadcast large meshes over low-bandwidth data connections like the Internet. Therefore the development of algorithms for progressively transmitting surface geometry is mandatory. The de facto standard for doing this in the case of triangle meshes is the technique of *progressive meshes* [6], where an initial, coarse shape is transmitted first and can be displayed immediately. With more detail information being received, the mesh can be progressively refined until the complete model is recovered.

In this paper we explore an alternative method for the progressive transmission of special triangle meshes which have the so-called *subdivision connectivity*. Meshes with subdivision connectivity are generated by many different algorithms where a uniform splitting operator is used for the refinement of an initial control mesh. By this refinement operator each triangle is divided into four by inserting one

* Corresponding author.
E-mail address: kobbelt@mpi-sb.mpg.de (L. Kobbelt).

0925-7721/00/$ – see front matter
PII: S0925-7721(99)00045-0

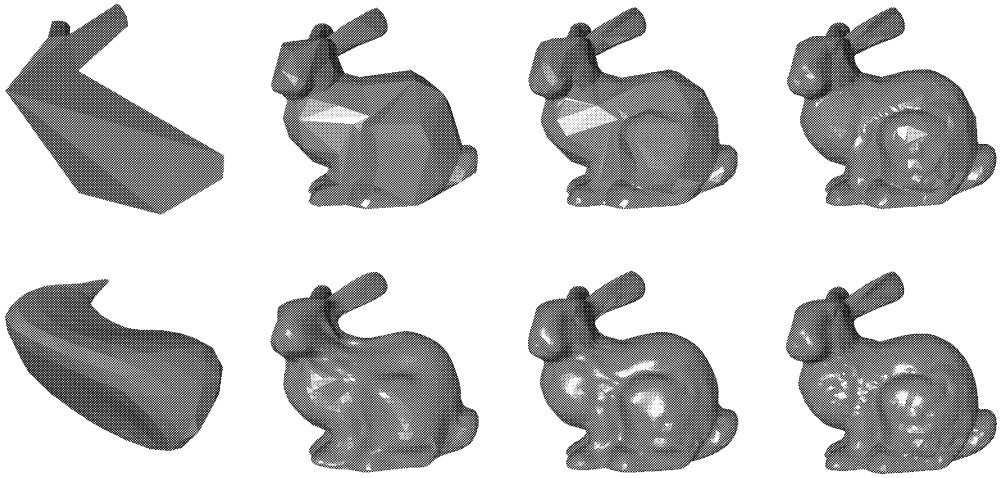


Fig. 1. Progressive meshes versus subdivision meshes. In the top row, a progressive mesh for the Stanford bunny is shown. From left to right: initial base mesh, 1%, 3% and 10% of the detail information received. The bottom row shows the progressive transmission of a (remeshed) subdivision representation of the bunny with the same percentage of included detail information.

new vertex per edge. Any given mesh with arbitrary connectivity can be converted into a mesh with subdivision connectivity within a prescribed approximation tolerance [4,8,9].

Besides the straight forward level of detail semantics, the use of subdivision meshes has several advantages. First, it is easy to generate smooth approximations of the original surface by applying a stationary subdivision rule on some coarser level of detail. In regions where no detail information is available, the vertex positions can be predicted by the subdivision operator. This is not possible for progressive meshes where only *coarse* approximations of the original mesh can be displayed but not *smooth* ones (see Fig. 1). The superior approximation power of subdivision basis functions compared to plain piecewise linear surfaces enables a significant reduction of the communication overhead while preserving the visual quality of the displayed object.

Since subdivision surfaces are naturally equipped with a global parameterization it is easy to identify the location of a detail coefficient by its dyadic barycentric coordinates within the associated base triangle. This indexing enables random access to the detail features in arbitrary mesh regions and on any level of detail. Notice that progressive meshes on the other hand are static in the sense that once the sequence of detail coefficients is generated, its ordering is more or less fixed due to mutual dependencies. Hence, a (real-time) multi-media communication protocol that cannot guarantee the correct ordering of the data packages is not appropriate for progressive meshes. Since there is not sufficient redundancy in the progressive mesh representation, one lost package during the transmission makes the reconstruction of the remaining detail information impossible.

The paper is organized as follows. After surveying the used subdivision schemes in Section 2 and discussing some related work in Section 3, we present a method for the decomposition of a given mesh by using inverse subdivision in Section 4. In Section 5 we define our representation for the detail coefficients during the progressive transmission of the mesh and in Section 6 we show an algorithm by which the receiver reconstructs the mesh from the transmitted data.

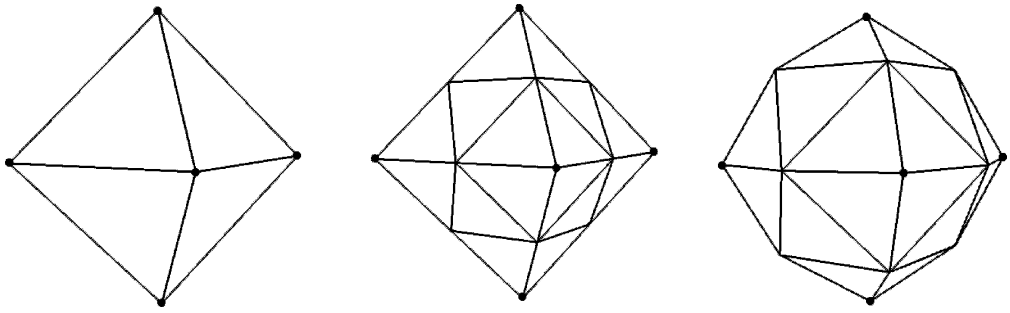


Fig. 2. Subdivision of an octahedron. Each face of the mesh is split into four triangles and the resulting mesh is smoothed by shifting the vertices. Even vertices in the refined mesh are marked by solid bullets.

2. Subdivision schemes

Subdivision schemes generate a sequence of successively refined meshes which converge to a smooth surface in the limit [13]. A uniform subdivision operator S refines a given mesh $\mathcal{M}^k$ to a new mesh $\mathcal{M}^{k+1}$. Each refinement step can be considered as the application of a topological 1-to-4 split operation followed by a smoothing operation that shifts the mesh vertices (cf. Fig. 2). Splitting each triangular face into four new triangles by inserting new vertices at the midpoints of the edges, increases the resolution and hence switches the geometry representation to the next level of detail. The smoothing operation computes the new vertex positions by fixed linear combinations of the neighboring vertices, which is called the stencil.

The subdivision process starts with an initial mesh $\mathcal{M}^0$. The set of vertices on the kth level of detail $\mathcal{M}^k$ are denoted by $\mathcal{P}^k$. By applying the subdivision operator

$$S : \mathcal{M}^k \to \mathcal{M}^{k+1} \tag{1}$$

refined meshes are generated. The mesh $\mathcal{M}^{k+1}$ has vertices $\mathcal{P}^{k+1}$ which can be classified in two different sets: the *even* vertices $\mathcal{P}^{k+1}_{\text{even}} = \mathcal{P}^k$ which correspond to vertices of the unrefined mesh $\mathcal{M}^k$, and the *odd* vertices $\mathcal{P}^{k+1}_{\text{odd}} = \mathcal{P}^{k+1} \backslash \mathcal{P}^k$ which are added in the current refinement step. The refined meshes $\mathcal{M}^k$ have subdivision connectivity generated by the uniform splitting operator.

For our implementation we use two different subdivision schemes, the Butterfly scheme [3,18] and Loop's scheme [10]. The two schemes differ in the way they compute the positions of the vertices on the next refinement level.

2.1. The Butterfly scheme

The Butterfly scheme is an interpolatory subdivision scheme. This means that the vertices of all refinement levels lie on the limit surface. Hence, when refining a triangle mesh, the even vertices do not change their position, i.e., the smoothing rule for the even vertices is simply the identity. The positions of the odd vertices $\boldsymbol{q}^{k+1}$ are computed by the following linear combination of neighboring vertices:

$$\boldsymbol{q}^{k+1} = \tfrac{1}{2}(\boldsymbol{p}_1^k + \boldsymbol{p}_2^k) + \tfrac{1}{8}(\boldsymbol{p}_3^k + \boldsymbol{p}_4^k) - \tfrac{1}{16}(\boldsymbol{p}_5^k + \boldsymbol{p}_6^k + \boldsymbol{p}_7^k + \boldsymbol{p}_8^k). \tag{2}$$

The location of the vertices $\boldsymbol{p}_i$ in Eq. (2) can be found in Fig. 3. The vertices in such a configuration used for subdivision schemes are called stencil points.

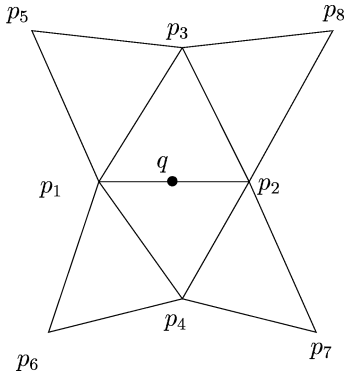


Fig. 3. Configuration of the stencil points in the Butterfly scheme.

2.2. Loop

In contrast to the Butterfly scheme, the Loop scheme is non-interpolatory, i.e., the even vertices change their position in the smoothing step. The position $\boldsymbol{p}^{k+1}$ is computed as a weighted sum of the corresponding vertex $\boldsymbol{p}^k$ from the coarser mesh and its direct neighbors $\boldsymbol{p}_i^k$:

$$\boldsymbol{p}^{k+1} = \frac{\alpha(v)}{\alpha(v)+v}\boldsymbol{p}^k + \frac{1}{\alpha(v)+v}\sum_i \boldsymbol{p}_i^k. \tag{3}$$

Here, v is the valence of $\boldsymbol{p}^k$. By the term valence we understand the number of neighbors connected to a vertex. The weight coefficients $\alpha(v)$ can be found in [10,19]. In the regular case $v = 6$ the value of $\alpha(v)$ is 10. In the general case, $v \neq 6$, the weight coefficient is defined as

$$\alpha(v) = v\frac{1-\beta(v)}{\beta(v)}, \qquad \beta(v) = \frac{5}{8} - \frac{(3+2\cos(2\pi/k))^2}{64}. \tag{4}$$

The positions for the odd vertices are calculated by

$$\boldsymbol{q}^{k+1} = \tfrac{3}{8}(\boldsymbol{p}_1^k + \boldsymbol{p}_2^k) + \tfrac{1}{8}(\boldsymbol{p}_3^k + \boldsymbol{p}_4^k). \tag{5}$$

2.3. Adaptive subdivision

Under uniform subdivision the number of faces grows exponentially with the refinement level. Although the iterative refinement always leads to a mesh with globally improved smoothness, in some areas with low curvature no significant improvement of the mesh quality is achieved by additional subdivision steps. Hence, subdivision should only be applied to faces which do not satisfy some prescribed *flatness criterion*; the other faces can remain coarse. This technique is called adaptive subdivision. It reduces the number of triangles in the resulting mesh and improves the rendering performance.

For the problem of progressive transmission we can use adaptive subdivision to refine the mesh only where detail information is inserted. Hence, the number of triangles stays as small as possible and the receiver achieves a higher performance when reconstructing and displaying the mesh.

Using adaptive subdivision is more difficult than uniform refinement since we have to avoid cracks in the mesh where faces from different refinement levels meet. In order to minimize the number of special

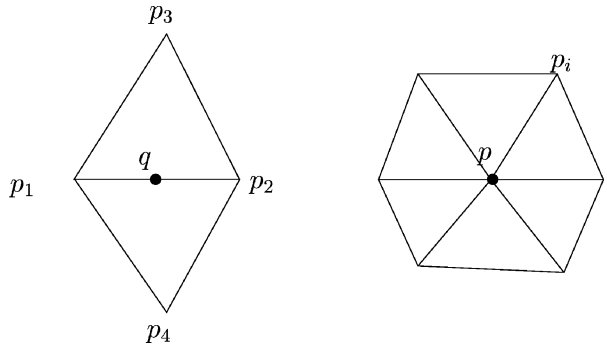


Fig. 4. Configuration of the stencil points in the Loop scheme.

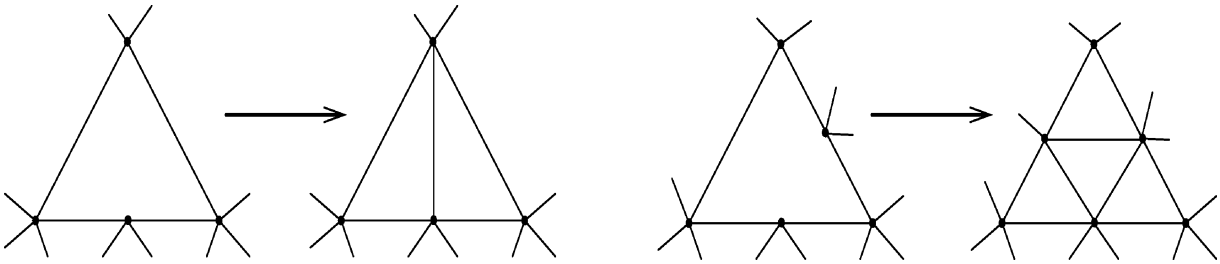


Fig. 5. In adaptively refined meshes, red splits fix the cracks where different refinement levels meet. If two neighbors of a triangle are split then the cracks are fixed by a green split.

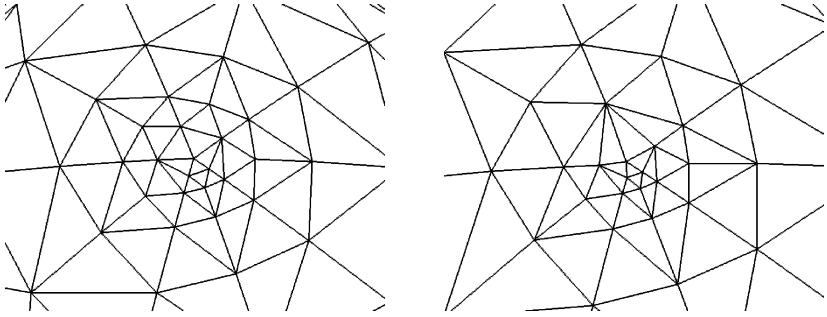
configurations we restrict the adaptivity to *balanced* meshes, i.e., the refinement level of two neighboring triangles may only differ by one.

For adaptive subdivision with a primal scheme like Loop or Butterfly, a special technique applies, the so-called *red–green triangulation* [16,17]. The ordinary 1-to-4 split of a face is called green split. To fix cracks, triangle bisection (red split) is applied if only one neighbor of a triangle is refined. If two or three neighbors are refined, a green split is applied which may cause further red splits of neighboring triangles. The red splits are only temporary. If a red split triangle is to be split further in a subsequent refinement step then the red split is undone first and then the green split is applied to the original triangle.

Another difficulty is the computation of the new vertices' position in an adaptively refined mesh because it can happen that some vertices in the stencil of the subdivision scheme have not been computed yet and that its computation requires further triangle splits.

In our adaptive refinement implementation we guarantee the balance of the refinement by requiring that the vertex which is associated with the midpoint of a certain edge can only be computed after both triangles adjacent to this edge (on the same refinement level) have already been generated. If one of the adjacent triangles is too coarse, an upward recursive splitting procedure is called.

If both adjacent triangles exist, the stencil vertices can be collected by using the neighborhood information stored in the mesh data structure. If a vertex is missing, it can be computed by recursively calling the same vertex evaluation procedure.



(a) (b)

Fig. 6. Adaptive subdivision. (a) The Butterfly scheme is used, (b) the Loop scheme is applied. Non-conforming cracks in the mesh are fixed by red–green triangulation.

ComputeEdgeVertex($face_1, face_2$)
  if $face_2.level < face_1.level$ then
    Refine ($face_2$)
    $face_2 = face_2.child[i]$ with $face_2.child[i]$ is neighbor of $face_1$
  $\forall \boldsymbol{p}_j \in$ stencil
    if $\boldsymbol{p}_j$ does not exist then
      ComputeEdgeVertex($face_x, face_y$) with $face_x, face_y$
      adjacent to the edge whose midpoint is $\boldsymbol{p}_j$
  $\boldsymbol{q} = S(\boldsymbol{p}_0, \ldots, \boldsymbol{p}_n)$
  return($\boldsymbol{q}$)

This procedure is called with the two faces defining the edge. The level of $face_1$ has to be higher or equal than the level of $face_2$. Otherwise the vertex which is to be computed would already exist. The vertices $\boldsymbol{p}_j$ are the vertices in the stencil of the applied subdivision scheme S. In our implementation we store one odd vertex for every edge of a triangle. Hence if just the vertex position is needed, no split has to be performed. In Fig. 6 the differences between the Butterfly scheme and the Loop scheme can be seen. Because of the larger stencil of the Butterfly rule, more triangle splits are initiated by the evaluation procedure.

3. Previous work

The contributions in the area of level of detail management for triangle meshes can be divided into two major categories, depending on the topology and connectivity of the regarded mesh.

For arbitrary connectivity meshes, Hoppe proposed in [6] a mesh representation which enables the progressive transmission of surface geometry. The idea is to store the sequence of edge collapses performed by a mesh decimation algorithm. If executed in reverse order, the corresponding vertex split operations progressively recover the original mesh. Starting with the coarsest mesh $\mathcal{M}^0$, each vertex split generates a refined approximation $\mathcal{M}^{i+1}$ from its predecessor $\mathcal{M}^i$. This hierarchical representation

allows any *level of detail* to be retrieved by iteratively increasing the accuracy of the mesh with an appropriate number of vertex split operations.

The big advantage of the progressive mesh representation is that it provides level of detail semantics for meshes with arbitrary connectivity. In [7] this hierarchy is even used for multi-resolution modifications of meshed surfaces. A disadvantage is that the coarse approximations may have severe artifacts due to bad tesselations and popping effects may be visible when inserting detail coefficients without the geomorph technique proposed in [6].

For progressive mesh compression other multi-resolution representations have been developed [2,14]. With these techniques an arbitrary connectivity mesh can be decomposed into a level-of-detail hierarchy. In contrast to the progressive meshes technique the information necessary to switch to a finer level can be stored more efficiently. This is done by using more complex refinement operations. The great advantage of progressive mesh compression techniques is the reduction of total data to be transferred. However, these methods share the disadvantages that only coarse approximations can be shown initially and there is no random access to detail information.

A completely different approach to the multi-resolution representation of triangle meshes is possible, when some restrictions on the mesh structure are imposed. *Subdivision connectivity* of parametric meshes enables the adaption of 2-dimensional wavelet techniques originally defined for the functional setting. Such wavelet based methods provide the hierarchical decomposition which is necessary for level of detail management. The adaption to meshes with subdivision connectivity was first proposed by Lounsbery et al. [11], where an algorithm to construct a multi-resolution analysis for subdivision surfaces is presented. The authors use the fact that there always exist hierarchical scaling functions which provide a representation of the limit surface. Since these hierarchical scaling functions satisfy a two-scale relation, the nested sequence needed in a multi-resolution decomposition is defined by using linear combinations of the subdivision scaling functions at each level. Since the resulting orthogonal wavelets would have global support, they give up the orthogonality and use pseudo-orthogonal wavelets, whose support can be prescribed in advance.

Based on such decompositions, the progressive transmission of surfaces is possible. After transmitting the base mesh of the subdivision connectivity mesh, the detail information is sent in the form of detail vectors. The actual geometry encoded by such detail vectors $\boldsymbol{d}$ is obtained by shifting points in a certain region of the mesh by some scalar multiple of $\boldsymbol{d}$. The modified region is determined by the support of the corresponding wavelet function. The scalar multiple is obtained by evaluating the wavelet function at the corresponding parameter value.

Using piecewise linear hat functions as scaling functions, this wavelet based approach is applied in [1], where the hierarchical representation is used for progressive transmission of colored meshes and viewing with constant frame rates. The influence of the wavelet support is investigated with the conclusion that increasing the size of the support does not significantly improve the approximation quality, but decreases the numerical stability and the performance. Hence, the authors recommend to chose the wavelet support as small as possible.

Another important step in the wavelet framework is done by [12], where the authors construct a multi-resolution analysis for data defined over the sphere. As in [11] they improve the properties of the wavelets, but instead of choosing pseudo-orthogonality they increase the smoothness and vanishing moments of the wavelet functions by applying the lifting scheme.

The multi-resolution analysis used for the progressive transmission in this paper is not based on explicit wavelet functions. Instead of investigating the theoretic approximation power of certain functional

bases, it is more important in our application to associate an intuitive geometric shape with each detail coefficient. Since the wavelet functions on level i are given by a linear combination of scaling functions from the next refinement level, we can express the detail information in terms of the coefficients for these scaling functions as well. The advantage of this representation is that the (difference) geometry associated with each coefficient inherits the intuitive bell-shaped distribution of the scaling functions. A disadvantage is that exact error measures like the L_2 norm are no longer available since orthogonality is lost. Hence, we have to find an alternative geometric criterion for sorting the detail coefficients according to their significance. As we will show in Sections 4 and 8, feature size and local approximation error provide good measures.

4. Decomposition

The first task for the progressive transmission of a given subdivision mesh $\mathcal{M}^n$ is the decomposition into a small base mesh $\mathcal{M}^0$ and a set of detail vectors which encode the geometric difference between the different refinement levels. The difference vectors can be sorted according some criterion which rates their significance. Simple heuristics which use the length of the vectors or the local approximation error with respect to the original mesh lead to feasible results.

To find the coarser approximations $\mathcal{M}^{n-1}, \ldots, \mathcal{M}^0$ we apply inverse subdivision, i.e., we undo the 1-to-4 splits on $\mathcal{M}^{k+1}$ and shift the remaining vertices of $\mathcal{M}^k$ in order to reduce the approximation error. Applying the stationary subdivision operator to the coarsified mesh $\mathcal{M}^k$ yields a smooth approximation $\widetilde{\mathcal{M}}^{k+1}$ of $\mathcal{M}^{k+1}$. The difference between the corresponding vertex positions is used for the detail vectors.

4.1. Inverse subdivision

The task of inverse subdivision is to transform a given mesh $\mathcal{M}^{k+1}$ into a coarser mesh $\mathcal{M}^k$ approximating the original. Obviously, inverse subdivision can only be applied if $\mathcal{M}^{k+1}$ has subdivision connectivity. Let S be a primal subdivision operator. Then a coarse approximation $\mathcal{M}^k$ of a mesh $\mathcal{M}^{k+1}$ has to satisfy

$$\mathcal{M}^{k+1} \approx S\mathcal{M}^k. \tag{6}$$

In the classical wavelet setting, the approximation is with respect to the L_2 norm defined over the base mesh $\mathcal{M}^0$ as the parameter domain. Since we want to encode the detail information in terms of displacement vectors multiplied by (scalar valued) scaling functions of the next refinement level, it makes more sense to rate the approximation according to the interpolation error at the even vertices of $\mathcal{M}^{k+1}$.

Hence, we have to assign positions to the vertices of the coarse mesh such that when refining the mesh again these vertices are mapped onto the even vertices of the original mesh, i.e.,

$$\mathcal{P}^{k+1}\big|_{\text{even}} = \big(S\mathcal{P}^k\big)\big|_{\text{even}}. \tag{7}$$

Due to the finite size of the subdivision rule's stencil, this is a sparse linear system which can be solved easily by some iterative scheme like Jacobi or Gauß–Seidel. In the case of the Butterfly scheme the solution is trivial since the even-rule for interpolatory schemes is the identity and hence $\boldsymbol{p}_j^k = \boldsymbol{p}_j^{k+1}$.

For Loop's scheme each row of the matrix has valence plus one non-zero entries (mostly 7) which can be read off from the even rule (3). The weak diagonal dominance of the resulting matrix guarantees stable convergence of any iterative solving algorithm.

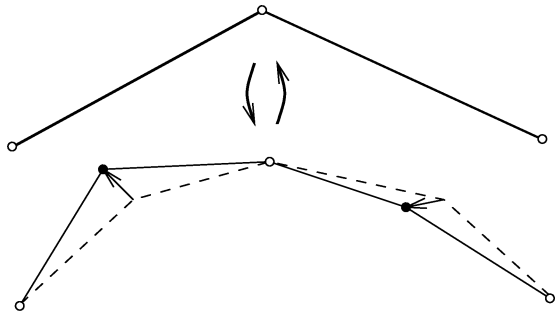


Fig. 7. The solid line below represents the original fine mesh $\mathcal{M}^{k+1}$, the solid line above the coarsified mesh $\mathcal{M}^k$. Applying the subdivision operator S to $\mathcal{M}^k$ yields the dashed line which interpolates $\mathcal{M}^{k+1}$ at the even vertices. By eventually adding the detail vectors to the odd vertices, the original mesh can be reconstructed.

When applying the subdivision operator S to the solution of (7), the even vertices of $\mathcal{M}^{k+1}$ are recovered but the predicted positions for the odd vertices have to be corrected in order to exactly reproduce the original data. Hence, we have to store a detail vector

$$\mathcal{D}^k := \mathcal{P}^{k+1}\big|_{\text{odd}} - \left(S\mathcal{P}^k\right)\big|_{\text{odd}} \tag{8}$$

for every odd vertex. During the reconstruction, these detail vectors are added after each subdivision step. The whole decomposition/reconstruction process is depicted in Fig. 7.

5. Representation

For the progressive transmission, we define a special encoding of the detail vectors. Each coefficient has to contain enough information such that the receiver can find the corresponding location in the mesh where the detail has to be included. Since we do not rely on some implicit information stored in the ordering of the detail coefficients, the sender can arbitrarily re-order the sequence of detail vectors without affecting the consistency of the surface reconstructed by the receiver.

The simplest way to represent the geometry of a plain polygonal mesh is to store a tuple (V, F), where V is a set of vertex positions and F is a set of faces, each defined by a list of references to the vertex list V. This is called the *shared vertex* representation.

After the decomposition of the given mesh $\mathcal{M}^n$, we use this representation for the base mesh $\mathcal{M}^0$. The progressive transmission of $\mathcal{M}^n$ starts by sending the base mesh encoded in shared vertex representation. The base mesh is followed by an appropriately sorted sequence of detail records.

Each record is given as a tuple $\{\boldsymbol{d}, n, p\}$ where $\boldsymbol{d} = [x, y, z]$ is the actual detail vector by which the corresponding vertex has to be shifted. n is the index of the base triangle T_n over which the detail is located. Each base triangle is the root node of an associated quad-tree that is generated by the topological 1-to-4 splits of the subdivision operator. The string p describes the path in the quad-tree that leads to the leaf where the detail vector $\boldsymbol{d}$ has to be included.

This path is build up as a sequence of symbols ‘0’, ‘1’, ‘2’ and ‘3’ defining the route within the quad-tree. The last symbol is a letter ‘A’, ‘B’ or ‘C’ indicating the particular corner vertex that has to be shifted by $\boldsymbol{d}$. Cf. Fig. 8 for an explanation of the symbol’s meaning. The circulant indexing is used to simplify

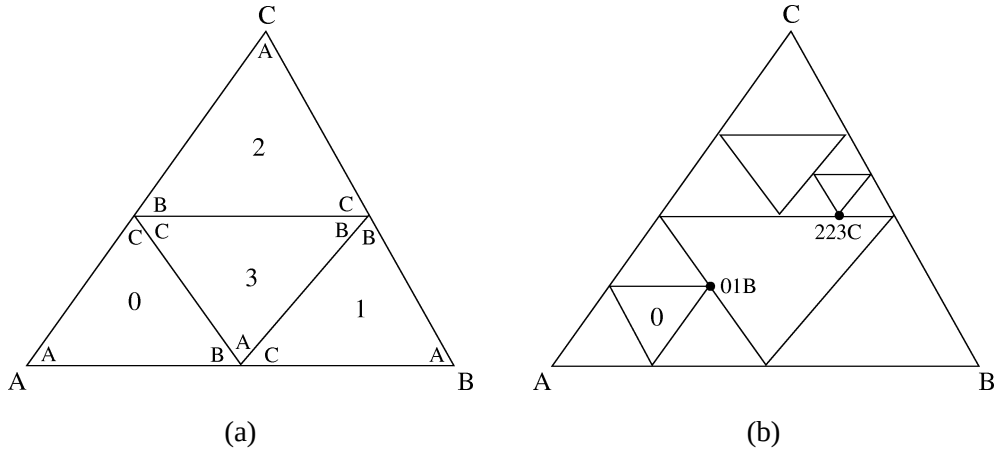


Fig. 8. (a) Indexing of the vertices and child triangles. (b) Geometric position of the vertices given in the example.

the implementation code since most special cases can be treated automatically by index computations *modulo* 3.

Here are some examples for the detail record encoding, the geometric positions of the addressed vertices can be found in Fig. 8.

```
0.1 0.2 0.3    12    01B
0.0 0.1 0.0    12    223C
```

It is obvious that there is some redundancy in this representation. There are many different paths to access the same vertex in the quad-tree structure. The memory requirements for storing a path is $O(n) = O(\log(m))$ with n the number of refinement levels of the mesh $\mathcal{M}^n$ and m the number of vertices in $\mathcal{M}^n$. From the mesh compression point of view, this is rather inefficient [2,5,14,15] but therefore the encoding allows random access to any detail vector. This is important for the progressive transmission since the receiver cannot make any assumptions on the order by which the server sends the detail records.

6. Reconstruction

The second task for the progressive transmission is the reconstruction of the mesh by the receiver. The detail records have to be inserted into the current approximation of the original mesh until it is completely reconstructed. During this process the viewer can display the progressively improving mesh. If the transmission is rather slow, it is reasonable to redraw the mesh after every newly received detail vector. If the transmission is sufficiently fast, blockwise redrawing with a fixed frame rate is more appropriate since it prevents another bottle-neck in the graphics pipeline.

The biggest problem for the mesh data structure underlying the reconstruction algorithm is to enable the processing of the detail records in any order. When starting with the coarse mesh $\mathcal{M}^0$ the vertices for which detail information is to be added may not exist and have to be created on the fly by applying the subdivision operator S to the mesh.

In our implementation the subdivision scheme is used adaptively. Hence the splitting of triangular faces is only performed where it is necessary. To improve the visual appearance of the mesh, subdivision

is always performed two levels finer than the current level of detail in that area. This means that the base mesh is initially subdivided two times before any detail is received. By this the user sees a smooth initial surface instead of a coarse initial mesh (cf. Fig. 1).

When a detail vector of level k is received, a small area around the corresponding vertex is subdivided to level $k+2$. Certainly, this is only done up to the maximum refinement level of the original mesh $\mathcal{M}^n$. Hence, when all detail records are processed, the original mesh is exactly reconstructed.

The justification for the constant refinement offset is that every feature is displayed by the same number of triangular faces. For high frequency detail from some higher level of detail these faces are smaller since the support of the associated subdivision basis function is smaller as well.

6.1. Inserting detail

The basic algorithm for inserting a detail vector is

InsertDetail (*detail*, *face*, *path*)
  $\boldsymbol{v}^k = \text{GetVertex}$ (*face*, *path*)
  RefineOneRing($\boldsymbol{v}^k, k$)
  $\boldsymbol{v}^k = \boldsymbol{v}^k + detail$
  UpdateVertices ($\boldsymbol{v}^k, k$)

Hence the first thing to do when inserting a detail record $\{\boldsymbol{d}, n, p\}$ into the mesh is to find the corresponding vertex addressed by the path p. Starting with the base triangle T_n, the algorithm recursively descends to the child triangle indicated by the next symbol in the path. If an addressed triangle does not exist (a leaf node of the quad-tree is reached), it is created by using adaptive subdivision. If the correct triangle is reached the requested vertex is chosen by the last symbol of p. The algorithm looks like this:

GetVertex(*face*, *path*)
  $k = path.length$
  $f = face$
  for ($i = 0$; $i < k-1$; $i++$)
    if $f.leaf$ then Subdivide(f)
    $f = f.child[path[i]]$
  return $f.vertex(path[k-1])$

Once the correct vertex $\boldsymbol{v}^k$ on level k is found, we refine the 1-ring around this vertex twice. This is done by first enumerating all triangles around vertex $\boldsymbol{v}^k$ and storing them in a list T. All faces in this list are refined twice if their children do not already exist and the maximum refinement level *maxdepth* is not exceeded.

RefineOneRing($\boldsymbol{v}, k$)
  $T = \text{GetOneRing}(\boldsymbol{v}, k)$
  $\forall f \in T$
    if $k < maxdepth$ then Refine f
    if $k < maxdepth - 1$ then Refine children of f

As mentioned earlier, this procedure guarantees a constant refinement offset, i.e., every detail feature is displayed with the same number of triangles no matter on which level the detail is inserted. After refining

the triangles in the 1-ring, the detail vector $\boldsymbol{d}$ is added to the chosen vertex $\boldsymbol{v}^k$. It is necessary to store the detail vector explicitly with the vertex because otherwise the information would get lost during an updating process triggered by the detail insertion to a neighboring vertex.

The local update process changes the position of all vertices within the support of the subdivision basis function associated with the vertex $\boldsymbol{v}^k$. Since this support differs for Loop and Butterfly subdivision, a case distinction has to be made for the implementation.

UpdateVertices$(\boldsymbol{v}, k)$
  if BUTTERFLY then $T = \text{GetThreeRing}(\boldsymbol{v}, k)$
  if LOOP then $T = \text{GetTwoRing}(\boldsymbol{v}, k)$
  while $T \neq \{\}$
    $f = T.head()$
    RecomputeVertices(f)
    Add detail to vertices
    if not $f.leaf$ then
      $T.append(f.child[i], i = 0, \ldots, 3)$

In this update process all vertices belonging to children of faces in T are recomputed. This means that the positions of these vertices are recomputed with the subdivision operator S applied to the already updated positions of the ancestors. In order not to loose the detail information added previously on higher refinement levels, these vectors must be added again to the associated vertices. The procedure updates the vertices level-wise by storing the children of a face at the end of the list T and fetching the first element from the list (FIFO-queue).

After all detail records are received and inserted the original mesh is reconstructed exactly, because no information got lost. However, it turns out that even with very few detail records received, a very good approximation of the original mesh is already given (cf. Section 9).

7. Progressive transmission

In the last sections we explained how to decompose and reconstruct a given mesh with subdivision connectivity and we specified a representation for storing that decomposition. Now we want to explain how to integrate these techniques for the progressive transmission of triangle meshes into a distributed system using general purpose computer networks like the Internet.

The progressive transmission is based on a client/server model (Fig. 9). The server task is to provide the clients with meshes stored in the progressive representation of Section 5. If a mesh has no subdivision connectivity it has to be remeshed first, i.e., the mesh is restructured and resampled in order to generate a subdivision connectivity mesh which approximates the originally given geometry up to a prescribed approximation tolerance [4,8,9].

The server acts like a remote library for geometric models which is accessed by distributed local clients. If the server receives a request from a client, it first transmits the base mesh of the requested model and after that the sequence of detail records is sent. The client task is to progressively reconstruct and display the mesh as explained in Section 6. The reconstruction algorithm is robust against packet loss during the transmission because of the random access to the mesh vertices and the independent indexing.

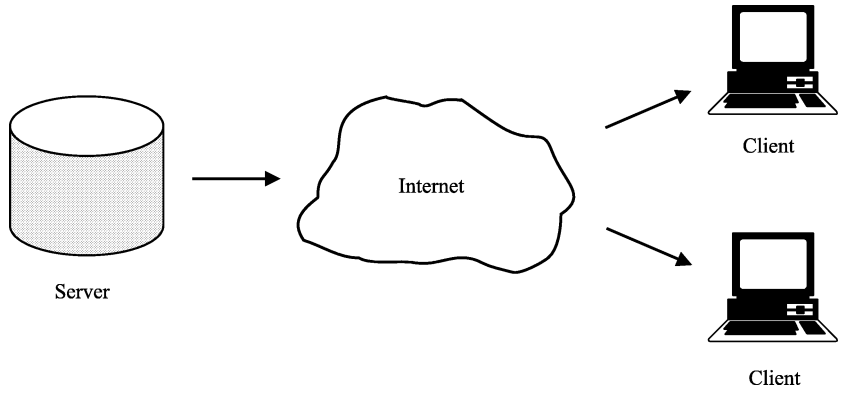


Fig. 9. Progressive transmission over the Internet.

When broadcasting animation sequences of polygonal models, the progressive transmission can be used to guarantee a constant frame rate. Every frame of the animation is offered by the server only for a certain period of time (determined by the target frame rate). After that period, the client displays the currently received model on the local screen and the transmission of the next frame is started. By this protocol the frames are broadcasted synchronously and varying load on the network results in varying geometric quality of the displayed model. This schedule enables the display of very complex animation sequences (like they are generated in simulation of mechanical deformation processes) on low cost client computers.

8. Results

We have applied our algorithm to a number of meshes. In Fig. 10 we show the progressive transmission of a brain model. The original model consists of 81920 triangles which can be reduced by decomposition to a base mesh of 20 triangles. The decomposition and reconstruction is done based on Loop's scheme. The other examples in Figs. 11 and 12 are produced by using the Butterfly scheme. The bust model consists of 73,728 triangles and is reduced down to a base mesh of 72 triangles. The well-known Spock-model has 32,768 triangles resulting from subdividing an octahedron.

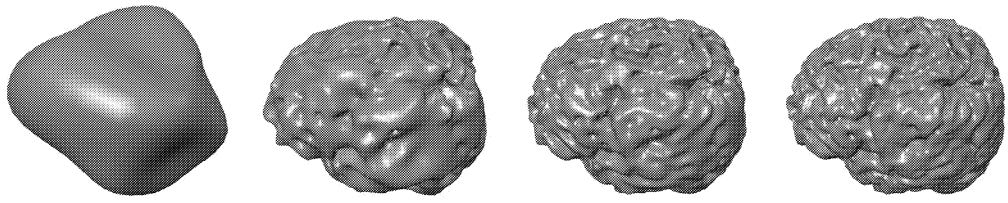


Fig. 10. Progressive transmission of the brain model. From left to right: 0%, 1%, 3% and 10% of the detail coefficients are included.

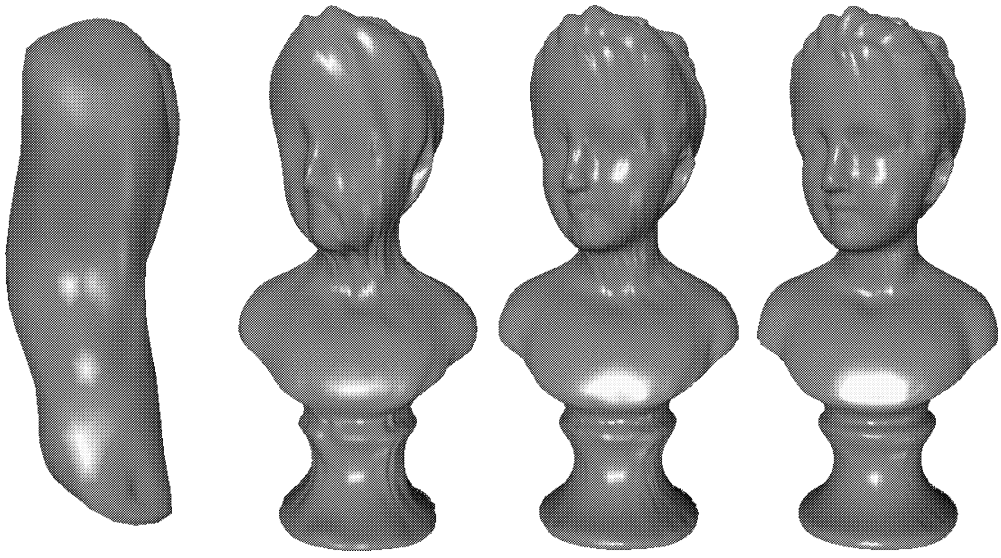


Fig. 11. Progressive transmission of the bust model. From left to right: 0%, 1%, 3% and 10% of the detail coefficients are included.

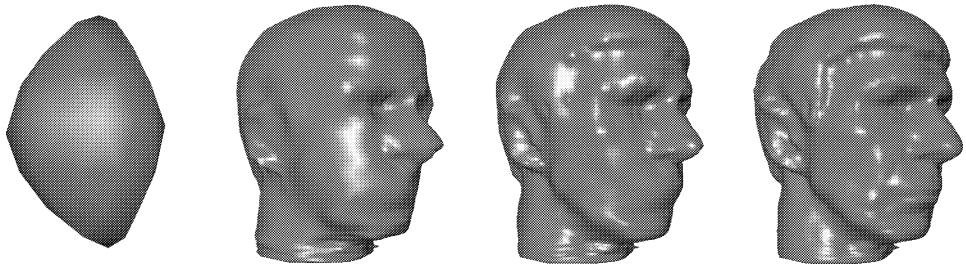


Fig. 12. Progressive transmission of the Spock model. From left to right: 0%, 1%, 3% and 10% of the detail coefficients are included.

9. Conclusions

We presented a very effective method for the progressive transmission of triangular meshes with subdivision connectivity. With our method it is possible to transform the representation of the given mesh into a progressive form with a small base mesh and a sequence of detail records. The detail records provide the information for the exact reconstruction of the original mesh. The advantage of our method is that these detail records can be processed in any given order. In contrast to other wavelet-based mesh compression schemes, we use only the coefficients of the scaling functions which provides an intuitive geometric interpretation for every detail vector. It turns out that already after a small amount of transmitted detail records is received, a very good approximation of the original data can be obtained. The number of triangles in the mesh during the reconstruction is reduced by using adaptive subdivision.

References

[1] A. Certain, J. Popovic, T. DeRose, T. Duchamp, D. Salesin, W. Stuetzle, Interactive multiresolution surface viewing, in: Computer Graphics (SIGGRAPH 96 Proceedings), 1996, pp. 91–98.
[2] D. Cohen-Or, D. Levin, O. Remez, Progressive compression of arbitrary triangular meshes, in: IEEE Visualization 99, 1999.
[3] N. Dyn, D. Levin, J. Gregory, A Butterfly subdivision scheme for surface interpolation with tension control, ACM Trans. Graphics 9 (1990) 160–169.
[4] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, W. Stuetzle, Multiresolution analysis of arbitrary meshes, in: Computer Graphics (SIGGRAPH 95 Proceedings), 1995, pp. 173–182.
[5] S. Gumhold, W. Straßer, Real time compression of triangle mesh connectivity, in: Computer Graphics (SIGGRAPH 98 Proceedings), 1998, pp. 133–140.
[6] H. Hoppe, Progressive meshes, in: Computer Graphics (SIGGRAPH 96 Proceedings), 1996, pp. 99–108.
[7] L. Kobbelt, S. Campagna, J. Vorsatz, H.-P. Seidel, Interactive multi-resolution modeling on arbitrary meshes, in: Computer Graphics (SIGGRAPH 98 Proceedings), 1998, pp. 105–114.
[8] L. Kobbelt, J. Vorsatz, U. Labsik, H.-P. Seidel, A shrink wrapping approach to remeshing polygonal surfaces, 1999, submitted.
[9] A. Lee, W. Sweldens, P. Schröder, L. Coswar, D. Dobkin, Multiresolution adaptive parametrization of surfaces, in: Computer Graphics (SIGGRAPH 98 Proceedings), 1998, pp. 95–104.
[10] C. Loop, Smooth subdivision surfaces based on triangles, Master's Thesis, Utah University, 1987.
[11] M. Lounsbery, T. DeRose, J. Warren, Multiresolution analysis for surfaces of arbitrary topological type, Technical Report 93-10-05, University of Washington, Department of Computer Science and Engineering, 1993.
[12] P. Schröder, W. Sweldens, Spherical wavelets: Efficiently representing functions on the sphere, in: Computer Graphics (SIGGRAPH 95 Proceedings), 1995, pp. 161–172.
[13] P. Schröder, D. Zorin, Subdivision for modeling and animation, in: Siggraph Course Notes 98, 1998.
[14] G. Taubin, A. Guéziec, W. Horn, F. Lazarus, Progressive forest split compression, in: Computer Graphics (SIGGRAPH 98 Proceedings), 1998, pp. 123–132.
[15] C. Touma, C. Gotsman, Triangle mesh compression, in: W. Davis, K. Booth, A. Fourier (Eds.), Proceedings of the 24th Conference on Graphics Interface (GI-98), San Francisco, 18–20 June 1998, Morgan Kaufmann, 1998, pp. 26–34.
[16] M. Vasilescu, D. Terzopoulus, Adaptive meshes and shells: Irregular triangulation, discontinuities, and hierarchical subdivision, in: Proceedings of Computer Vision and Pattern Recognition Conference, 1992, pp. 829–832.
[17] R. Verfürth, A Review of a Posteriori Error Estimation and Adaptive Mesh Refinement Techniques, Wiley/Teubner, 1996.
[18] D. Zorin, P. Schröder, W. Sweldens, Interpolating subdivision for meshes with arbitrary topology, in: Computer Graphics (SIGGRAPH 96 Proceedings), 1996, pp. 189–192.
[19] D.N. Zorin, Subdivision and multiresolution surface representations, Ph.D. Thesis, Caltech, Pasadena, CA, 1997.